

REMARKS

Claims 1-30 are pending in the present application. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 102, Alleged Anticipation, Claims 1-8, 11-18 and 21-28

The Office Action rejects claims 1-8, 11-18 and 21-28 under 35 U.S.C. § 102(e) as being allegedly anticipated by Crelier (U.S. Patent No. 6,151,703). This rejection is respectfully traversed.

As to claims 1, 11 and 21, the Office Action states:

As per claim 1, Crelier teaches the invention as claimed, including a method of calling a portion of computer code in a multithreaded environment, comprising:

receiving a call to the portion of computer code (col. 4 lines 8-35);

determining if the portion of computer code is currently being compiled (col. 12 line 57 – col. 13 line 3); and

redirecting the call to an interpreter, if the portion of computer code is currently being compiled (col. 12 line 57 – col. 13 line 3).

Office Action dated April 29, 2004, page 2.

Claim 1, which is representative of the other rejected independent claims 11 and 21 with regard to similarly recited subject matter, reads as follows:

1. A method of calling a portion of computer code in a multithreaded environment, comprising:

receiving a call to the portion of computer code;

determining if the portion of computer code is currently being compiled; and

redirecting the call to an interpreter, if the portion of computer code is currently being compiled. (emphasis added)

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. In re Bond, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. In re Lowry, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034

(Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicants respectfully submit that Crelier does not identically show each and every feature arranged as they are in the claims. Specifically, Crelier does not teach determining if the portion of computer code is currently being compiled and redirecting the call to an interpreter, if the portion of computer code is currently being compiled.

Crelier is directed to a development system having a client which employs a virtual machine for executing programs written in the Java programming language. The client executes a compiled Java program, which has been created by compiling a Java source code program or script with a Java compiler. The pseudo-compiled program comprises the bytecode emitted by the compiler. The development system further includes a just-in-time compiler which natively compiles each pseudo-compiled method of a Java program on a "just-in-time" basis. Methods which are unused are left uncompiled (i.e., left as bytecode). During program execution, when a method call is made from interpreted code, the system employs an "invoker" slot of the callee. When a method call is made from compiled code, the system employs a "compiled code" slot of the callee. As the addresses for the slots themselves remain unchanged, a method which has been compiled need not be recompiled when a callee method it invokes is itself compiled. In this manner, a method (caller) calling another method (callee) need not know whether the method it is calling is an interpreted method or a compiled method.

Thus, with the system of Crelier, the just-in-time compiler compiles each method as the method is actually used and methods which are unused are left uncompiled. Each method is compiled upon its first invocation, even though a method may execute only once and all subsequent uses of the method invokes the compiled version of the method. There is no teaching anywhere in the Crelier reference as to determining if the portion of computer code is currently being compiled. The Office Action alleges that this feature is taught by Crelier at column 12, line 57 to column 13, line 3, which reads as follows:

At initialization, the compiled code slot stores a callback pointer into the interpreter, since the callee method is not yet compiled at that point. When calling from compiled code to non-compiled code, the system encounters two levels of indirection. First, the compiled code calls back

into the Java interpreter (via one of the above callbacks). Second, the interpreter calls into the non-compiled method using the invoker slot, which can, in turn, invoke `invokeCompiler` (in the manner previously described).

In this section, Crelier is describing a technique used when the method associated with the method block or "callee" has not been compiled. Once a determination has been made that the method has not been compiled the previously compiled code calls back to the Java interpreter and the interpreter calls into the non-compiled method using the invoker slot, which can, in turn, invoke the `invokeCompiler` and invoke the method to be compiled. Thus, the system of Crelier is concerned with determining if a method has previously been compiled and, if not, compiling the method. Crelier is not concerned with determining if the portion of computer code is currently being compiled in a multithreaded environment. That is, Crelier compares incoming method calls to the previously compile code slot in memory, not what is currently being compiled in the processor.

Crelier also does not redirect the call to an interpreter, if the portion of computer code is currently being compiled. Crelier merely teaches that in the event that the caller method is previously compiled code, the compiled code slot is employed and if the callee method is also compiled, the compiled code slot stores a pointer to a memory block which comprises the just-in-time compiled code for the method, which is an optimization that allows direct invocation of the callee method's compiled code from the caller's compiled code, without having to go back through the interpreter or through a stub. As described above, in the instance where the callee method is not compiled, the compiled code slot stores a callback stub or wrapper function in order to jump back into previously interpreted code. Once a determination has been made that the method has not been compiled, the previously compiled code calls back to the Java interpreter and the interpreter calls into the non-compiled method using the invoker slot, which can, in turn, invoke the `invokeCompiler` and invoke the method to be compiled. Thus, Crelier does not teach redirecting the call to an interpreter, if the portion of computer code is currently being compiled. To the contrary, Crelier teaches redirecting the call to an interpreter only if the code has not been previously compiled.

Thus, Crelier does not teach each and every feature of independent claims 1, 11 and 21 as is required under 35 U.S.C. § 102(e). At least by virtue of their dependency on independent claims 1, 11 and 21, the specific features of dependent claims 2-8, 12-18 and 22-28 are not taught by Crelier. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1-8, 11-18 and 21-28 under 35 U.S.C. § 102(e).

Furthermore, Crelier does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. Absent the Examiner pointing out some teaching or incentive to implement Crelier to determine whether the portion of computer code is currently being compiled and redirect the call to an interpreter if the portion of computer code is currently being compiled in a multithreaded environment, one of ordinary skill in the art would not be led to modify Crelier to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify Crelier in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the Applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

Moreover, in addition to their dependency from independent claims 1, 11 and 21, respectively, Crelier does not teach the specific features recited in dependent claims 2-8, 11-18 and 21-28. For example, with regard to claims 3, 13 and 23, Crelier does not teach where in the step of redirecting the call to an interpreter, which is performed if the portion of computer code is currently being compiled, includes redirecting the call to a Java Virtual Machine Interpreter such that the portion of computer code is interpreted by the Java Virtual Machine Interpreter in response to receiving the call to the portion of computer code. The Office Action alleges that this feature is taught at column 12, line 57 to column 13, line 3, shown above. While Crelier may teach that a Java Virtual Machine may use a Java interpreter, Crelier teaches away from using a Java interpreter by implementing a system where methods are compiled, stored and compared to incoming method calls and if the comparison is identical, using the previously compiled method. Thus, Crelier does not teach or fairly suggest the specific features as recited in claims 3, 13 and 23.

As an additional example, with regard to claims 4, 14 and 24, Crelier does not teach wherein determining if the portion of computer code is currently being compiled

includes determining a setting of a flag in a control block of the portion of computer code. The Office Action alleges that this feature is taught at column 11, lines 19-40, which reads as follows:

In accordance with the present invention, the method blocks of methods are modified for use with the just-in-time compiler, as shown in FIG. 5. For method block 560, for example, use of its two slots are adapted as follows. First, the invoker slot 564 (corresponds to invoker 464 of FIG. 4) is employed from all calls from an interpreted caller (or the runtime interpreter) to the callee method (i.e., the method associated with the method block 560). When the method is compiled, the invoker slot 564 is updated so that, for subsequent calls, the compiled code for the method is executed. Ordinarily, the invoker slot stores a pointer to one of the following callback stub functions: invokeJavaMethod, invokeSynchronizedJavaMethod, invokeAbstractMethod, or invokeLazyNativeMethod. This is extended to also include the following new stub functions: invokeCompiler and invokeCompiledMethod. The invokeCompiler stub address is stored in the invoker slot when the method has yet to be compiled (and compilation has not be disabled). After a method has been compiled, the slot stores the address of the invokeCompiledMethod stub. Thereafter, calls from an interpreted caller to the method will results in invocation of the compiled version of the method, via the invoke CompiledMethod stub.

As shown previously, Crelier does not determine if the portion of computer code is currently being compiled. Furthermore, this section of Crelier clearly states that when the method is compiled, the invoker slot is updated so that, for subsequent calls, the compiled code for the method is executed. While Crelier may teach setting an invoker slot, it is not performed in determination that the portion of computer code is currently being compiled. Thus, Crelier does not teach or fairly suggest the specific features as recited in claims 4, 14 and 24.

As a further example, with regard to claims 6, 16 and 26, Crelier does not teach determining if compilation of the portion of computer code has ended and redirecting the call to a compiled version of the portion of computer code if the compilation of the portion of computer code has ended. The Office Action alleges that this feature is taught by Crelier at column 12, lines 31-40, which reads as follows:

When a method is called at runtime through the invoker slot (i.e., the caller is interpreted method or the Java runtime interpreter), the invokeCompiler stub is called to compile the method (since, at this point, the invokeCompiler pointer is stored in the invoker slot). This leads to

invocation of the goOnInvoke function, with the address of the just-compiled code. Afterwards, the invokeCompiledMethod pointer is stored in the invoker slot; compiled code slot is updated with a pointer to the compiled code. Subsequent calls to the method (from a non-compiled caller) will invoke the compiled code via the invokeCompiledMethod stub.

This section of Crelier merely teaches compiling of uncompiled code. It further teaches that once the code is compiled it is updated with a pointer to the compiled code and that subsequent calls to the method will invoke the compiled code. Thus, all future calls to the method are directed to the previously compiled code. Nowhere in this section, or any other section of Crelier, is it taught to determine if compilation of the portion of computer code has ended and redirect the call to a compiled version of the portion of computer code if the compilation of the portion of computer code has ended. Thus, Crelier does not teach the specific features recited in claims 6, 16 and 26.

Therefore, in addition to being dependent on independent claims 1, 11 and 21, respectively, dependent claims 2-8, 12-18 and 22-28 are also distinguishable over Crelier by virtue of the specific features recited in these claims. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 2-8, 12-18 and 22-28 under 35 U.S.C. § 102(e).

II. 35 U.S.C. § 103, Alleged Obviousness, Claims 9-10, 19-20 and 29-30

The Office Action rejects claims 9-10, 19-20 and 29-30 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Crelier (U.S. Patent No. 6,151,703). This rejection is respectfully traversed.

Claims 9, 10, 19, 20, 29 and 30 are dependent on independent claims 1, 11 and 21 and, thus, these claims distinguish over Crelier for at least the reasons noted above with regard to claims 1, 11 and 21. That is, Crelier does not teach determining if the portion of computer code is currently being compiled and redirecting the call to an interpreter, if the portion of computer code is currently being compiled. Moreover, the Office Action may not use the claimed invention as an "instruction manual" or "template" to piece together the teachings of the prior art so that the invention is rendered obvious.

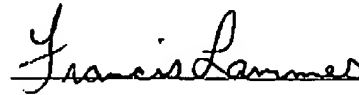
In view of the above, Crelier fails to teach or suggest the specific features recited in independent claims 1, 11 and 21, from which claim 9, 10, 19, 20, 29 and 30 depend. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 9, 10, 19, 20, 29 and 30 under 35 U.S.C. § 103.

III. Conclusion

It is respectfully urged that the subject application is patentable over the prior art of record and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: June 23, 2004



Francis Lammes
Reg. No. 55,353
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 367-2001
Agent for Applicants